daVinci:
eBox 4864 – Sentalis Fetch CSV Server
(Final Draft)

Christopher Steiner

Dr. Janusz Zalewski

CEN 4935 Spring 2011 – Senior Software Engineering Project

Florida Gulf Coast University

Fort Myers, Florida

4-30-11

1. Introduction

1.1 Project Overview

The purpose of this project is to create a dedicated server for downloading data from the Florida Gulf Coast University's Solar Plant's website SolarEMS. This project uses an eBox-4864 Embedded PC and utilizes Ubuntu 10.10 Linux operating system. This unit downloads the data, which are stored in comma separated value (CSV) files from the SolarEMS website using the Java driver software, HTMLUnit. The system, named daVinci, stores these files locally on its hard disk and share the information with the group's server in order for the other parts of the group project to access the data. It also stores it on the group server as a backup. In essence, daVinci is a data acquisition unit for the solar plant.

1.2 Project Materials

In this section the materials required in completing this project are listed and elaborated on. Also, other non-essential resources are listed.

1.2.1 Required Materials

- ICOP eBox-4864 Embedded PC (Figure 1.1)
- Computer with the following specifications
  - Intel Core Duo Processor
  - 1GB of RAM
  - Windows XP Professional

- At least 1 USB Port
- CD-ROM Drive

- Ubuntu 10.10 Minimal CD Image [9]
- 2 USB Flash Drives (at least 1GB in size)
- 4GB Compact Flash Card

1.2.2 Additional Resources

This list of additional resources consists of printed material and websites that furthered the understanding of the eBox-4864 with Ubuntu 10.10 and turning the eBox into a data acquisition unit and file server.

- eBox-4864 User Manual [1]
- Eclipse Classic 3.6.1 IDE [2]
- Java Runtime Environment Version 6 Update 24 [3]
- HTMLUnit [4]
- UNetBootIn [5]
- GParted [6]
- Samba [7]

1.3 eBox-4864 Components

The eBox-4864 is a fan-less embedded PC that uses a Via Esther 1.2GHz processor and 1GB of RAM. This embedded PC has 6 USB ports, a VGA port, PS2 keyboard and mouse ports, a network interface card, a wireless adapter and a compact flash port. (Figure 1.2)

Figure 1.1 Front Side of eBox-4864

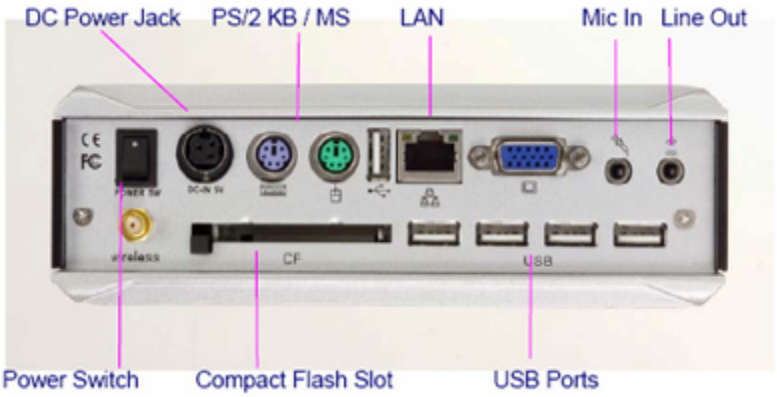


Figure 1.2 Back Side of eBox-4864.

2. Problem Description

2.1 Project Overview

The project uses an embedded PC, the eBox-4864, to create a central data acquisition server for the group project for Senior Software Engineering Project CEN 4935. The way this is accomplished is to load Ubuntu 10.10 onto the eBox to take full functionality of this board's network and compact size. The operating system is loaded onto a 4GB compact flash card via USB Flash Drive installation of Ubuntu 10.10. The program is written in Java using HTMLUnit, a Java unit testing framework for simulating a browser for testing purposes. The point of this project is to download the CSV files from the SolarEMS website [10] and store them locally and externally for redundancy. daVinci is able to hold approximately 67 years of data from the Solar Plant sensors via CSV files. The way that this is calculated is that each CSV file is approximately 150KB of data. Ubuntu uses approximately 500MB of space. This leaves 3.5GB of space left on the 4GB compact flash card. CSV files are stored once a day and just rewritten throughout the day so there is one CSV file per day, which is 365 CSV files a year. Which is 54,750KB a year, which means that daVinci can store 67 years of data before the 3.5GB compact flash card is full.

2.1.1 Assumptions

- daVinci operates on an eBox-4864 running Ubuntu 10.10.
- The user understands the functionality of daVinci and is able to use it based on documentation.
- Once the system is loaded on the eBox-4864 it does not need to be reloaded.

- daVinci effectively downloads and stores the CSV files locally and on a shared drive on the group's server.

2.1.2 Event Table

Event Name	Event Stimuli	External Responses	Internal data & state
daVinci is Turned On	Sentalis Fetch CSV program starts	"Currently Downloading CSV"	Store CSV file locally in a shared folder and externally in shared drive on group server.

2.2 Specific Requirements

2.2.1 Functional Requirements

- Once daVinci is started, it will start downloading CSV files.
- It will do this every hour until it is stopped or a system failure occurs.
  - Failures for this system are:
    - Internet connection is lost.
    - Misreading of the HTML website happens, causing an I/O Exception
    - SolarEMS website is down.
- The sequence of operations should follow:
  - daVinci gathers the CSV File from the SolarEMS website
  - daVinci stores this into a local folder that is shared called `shareddavinci`
  - daVinci stores this into an external shared drive on the groups server.
  - daVinci will wait one hour and then restart this process.

2.2.2 Interface Requirements

- Once daVinci is running there is no need for interface directly with daVinci.

- The only interface with daVinci are the CSV files located in the shared drive on daVinci's harddrive.

2.2.3 Physical Environment Requirements

- The software runs on the eBox-4864.
- As long as the eBox has network connection it should be able to accomplish its available tasks.

2.2.4 Users and Human Factors Requirements

- The types of users that the system will support are anyone interfacing with the shareddrive on daVinci.
- Each user must read instructions from **4.3 Installation** in order to know how daVinci works.
- The system automatically stores information as designed by developer.

2.2.5 Documentation Requirements

- There is a set of directions [Appendix] for the user to understand how daVinci works.
- This user manual is to instruct the user on how to properly connect to daVinci.

2.2.6 Data Requirements

- The data are be collected with HTMLUnit framework.
- Data generated by daVinci are output to shared folders and drives.

3. Solution

3.1 Solution Plan

The plan for creating a solution to the problem described in Section 2.1 was a five phase plan of action.

- Phase 1 – The compact flash card is inserted into daVinci. GParted is downloaded and setup onto one of the USB Flash Drives. Once it is installed, it will be plugged into daVinci and daVinci will be set to boot from this USB Flash Drive. GParted is ran on the CF Card in order to create the correct partition. Once GParted is ran successfully on the CF Card and the partition is created, Phase 2 can start. Phase 1 is discussed in more detail in the Section 3.2 (Solution Design).
- Phase 2 – With the partition created, Ubuntu 10.10 minimal cd image is downloaded and set to boot from USB using the second USB Flash Drive and UNetBootIn. When the USB Flash Drive is created, it is plugged into daVinci. daVinci is set to boot from this USB Flash Drive. Ubuntu is then installed via USB Flash Drive. Once this is completed, Phase 3 can start. Phase 2 is discussed in more detail in the Section 3.2 (Solution Design).
- Phase 3 – With Ubuntu successfully installed on daVinci, the environment is set up in order to run the Sentalis Fetch CSV program. The Java environment is downloaded and installed. Samba [7] is also downloaded and installed. Phase 3 is discussed in more detail in the Section 3.2 (Solution Design).

- Phase 4 – Once the environment has been successfully set up on daVinci a shared folder is created on daVinci to store the CSV files. Once this has been set up phase 5 can begin. Phase 4 is discussed in more detail in the Section 3.2 (Solution Design).
- Phase 5 – The developed program is then loaded into daVinci. After the developed Sentalis Fetch CSV program been loaded successfully into daVinci, testing can commence. The testing plan will has been developed and each individual test that needs to be run to achieve optimal functionality is to be ran in sequential order to ensure that daVinci is working properly. The test plan is to be updated in the case that there are unforeseen problems that arise during the testing phase. Phase 5 is discussed in more detail in the Section 3.2 (Solution Design).

3.2 Solution Design

3.2.1 Diagrams

This section shows a conceptual diagram in Figure 3.1 and a data flow diagram in Figure 3.2.

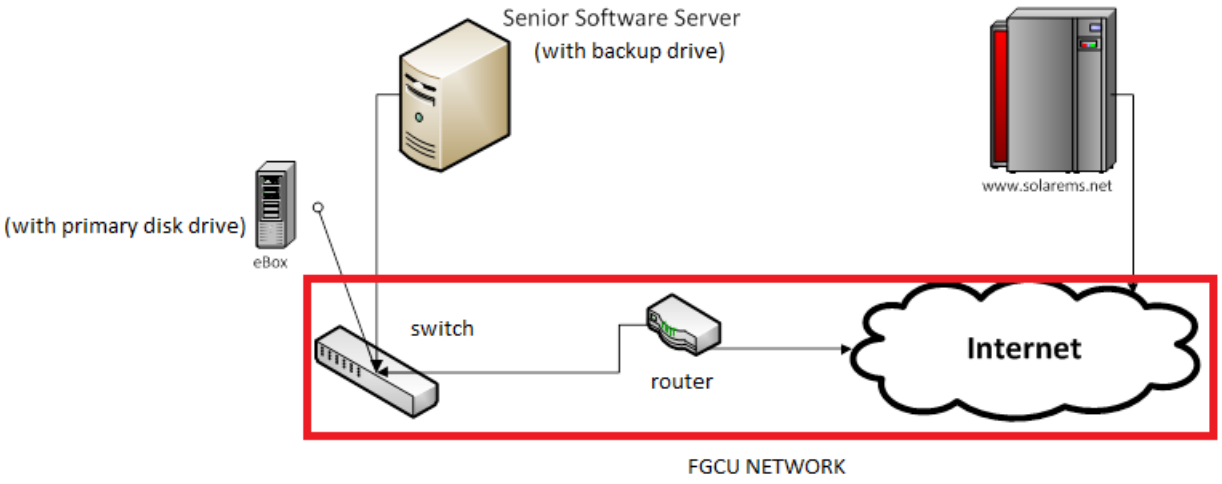
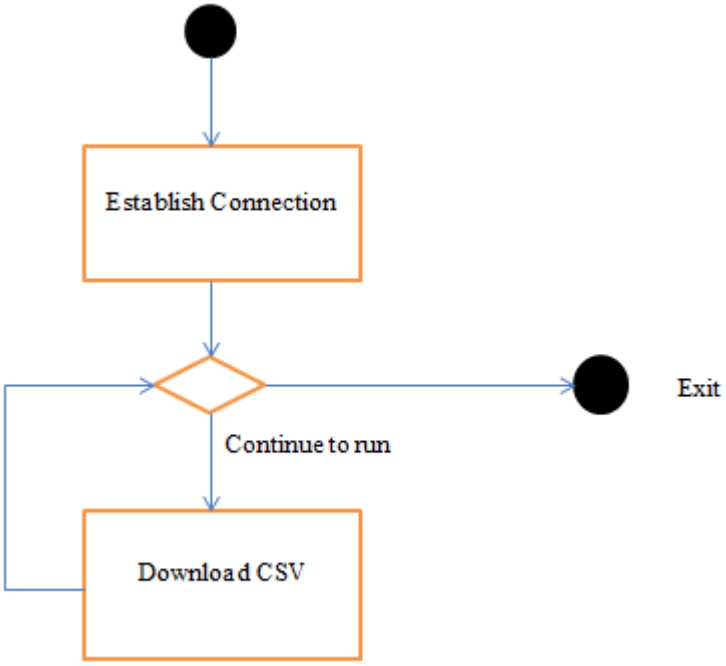


Figure 3.1 Physical Diagram of the System



(Figure 3.2) Data Flow Diagram of Program Operation

3.2.2 Test Plan

3.2.2.1 Description of Test Environment

The test environment is the daVinci Server. The eBox is loaded with the Sentalis Fetch CSV Program and is started to ensure the server starts properly. The test environment should exactly simulate the environment in which the software will ultimately operate.

3.2.2.2 Stopping Criteria

If an error occurs during any test case, the testing phase will continue until daVinci functions optimally. The subsequent result of the test will then be recorded and documented accordingly. The test result documentation is updated for each iteration of the test phase, per test case.

If the error occurs during download of CSV files, the error will be recorded, documented and the project returns to development phase in order to ensure that the Java code operates to design specifications.

The system is deemed complete when each test phase is completed with no errors. Once this happens, the documentation is updated with the final test results and the system can then be demonstrated.

3.2.2.3 Description of Individual Test Cases

- Test Case 1
  - Test Objective – Ensure that daVinci turns on properly.
  - Test Description – Turn the daVinci power switch to the on position.
  - Test Conditions – This test should be completed before any other test cases start as this is a vital part of the testing phase.
  - Expected Results – Once daVinci is powered on Ubuntu should load to show that daVinci is ready for use.
- Test Case 2
  - Test Objective – Run the Sentalis Fetch CSV Java code from the command prompt.
  - Test Description – When the Java code is executed daVinci should start downloading the csv files.

- Test Conditions – This test case should be executed for a few hours to ensure the csv files are downloading correctly.
- Expected Results – csv files should be output to the shared folder and shared drive.

- Test Case 3
  - Test Objective – Fill hard drive and try downloading again.
  - Test Description – The hard drive of daVinci will be filled with useless data to where it cannot hold another csv file.
  - Test Conditions – This test is to determine what will happen when daVinci finally reaches its capacity.
  - Expected Results – daVinci will fail as this far exceeds its limits.

3.3 Current Solution Results

Currently, the code has been written for Sentalis Fetch CSV. It works the way that it is intended to work. The environment is setup on daVinci and the local shared folder is available. The CSV Java code works in the Ubuntu environment and the jar file has been installed onto daVinci. It is currently running once every hour and placing the .csv files into a local shared drive and over the network onto the backup location on the main server.

4. Implementation

To run the daVinci software, you need to make sure that the hardware is connected as in Figure 3.1, the eBox has Ubuntu 10.1 operating system installed, it runs Samba and the shared drive on the Senior Software server is mounted. Refer to Section 4.3 for additional information.

After power-up the login screen appears, login with username and password. At this point we need to mount the Senior Software Server shared drive using the following command:

```
sudo mount –t cifs username=<username>,password=<password> //backupserver/backups
/mnt/win
```

After the drive is mounted then run:

```
sudo java –jar SentalisFetchCSV.jar
```

This will start the daVinci software and download the CSV files once every hour. There are many warnings for JavaScript and CSS as the htmlunit platform does not support these things and SolarEMS uses them to route traffic and show their website with CSS. These warnings can be ignored.

4.1 Test Results

4.1.1 Test Case 1 Results

- Test Objective – Ensure that daVinci turns on properly.
- Test Description – Switch the daVinci power switch to the on position.

- Test Conditions – This test should be completed before any other test cases start as this is a vital part of the testing phase.
- Expected Results – Once daVinci is powered on Ubuntu should load to show that daVinci is ready for use.
- Actual Results – daVinci powered on fine with no issues. Ubuntu loaded as intended.

4.1.2 Test Case 2 Results

- Test Objective – Run the Sentalis Fetch CSV Java code from the command prompt.
- Test Description – When the Java code is executed daVinci should start downloading the csv files.
- Test Conditions – This test case should be executed for a few hours to ensure the csv files are downloading correctly.
- Expected Results – csv files should be output to the shared folder and shared drive.
- Actual Results – The Sentalis Fetch CSV Java code executes as intended and stores the csv files locally in the shared folder and externally on the shared drive.

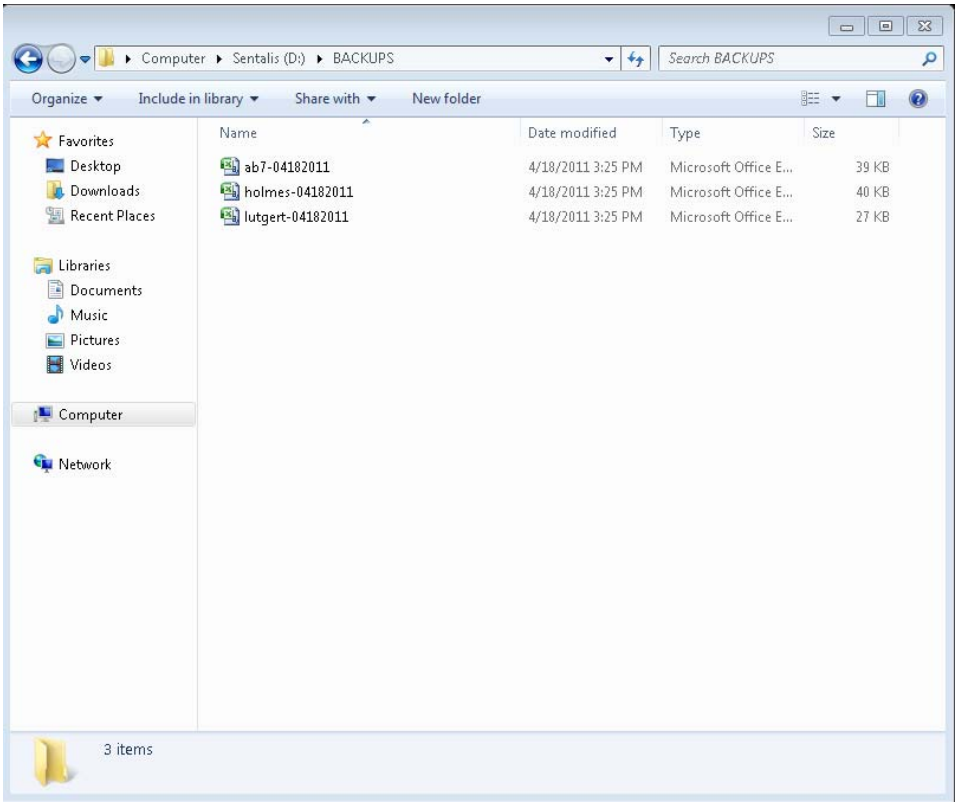


Figure 4.1 Sentalis Shared Drive on Group Server

4.1.3 Test Case 3 Results

- Test Objective – Fill hard drive and try downloading again.
- Test Description – The hard drive of daVinci will be filled with useless data to where it cannot hold another csv file.
- Test Conditions – This test is to determine what will happen when daVinci finally reaches its capacity.
- Expected Results – daVinci will fail as this far exceeds its limits.
- Actual Results – daVinci was unable to write the CSV files to their appropriate locations.

4.2 Operation and Performance

Each iteration of testing that was completed showed correct results. daVinci works as designed each and every time of use. Each hour daVinci successfully downloads and stores the CSV files as intended.

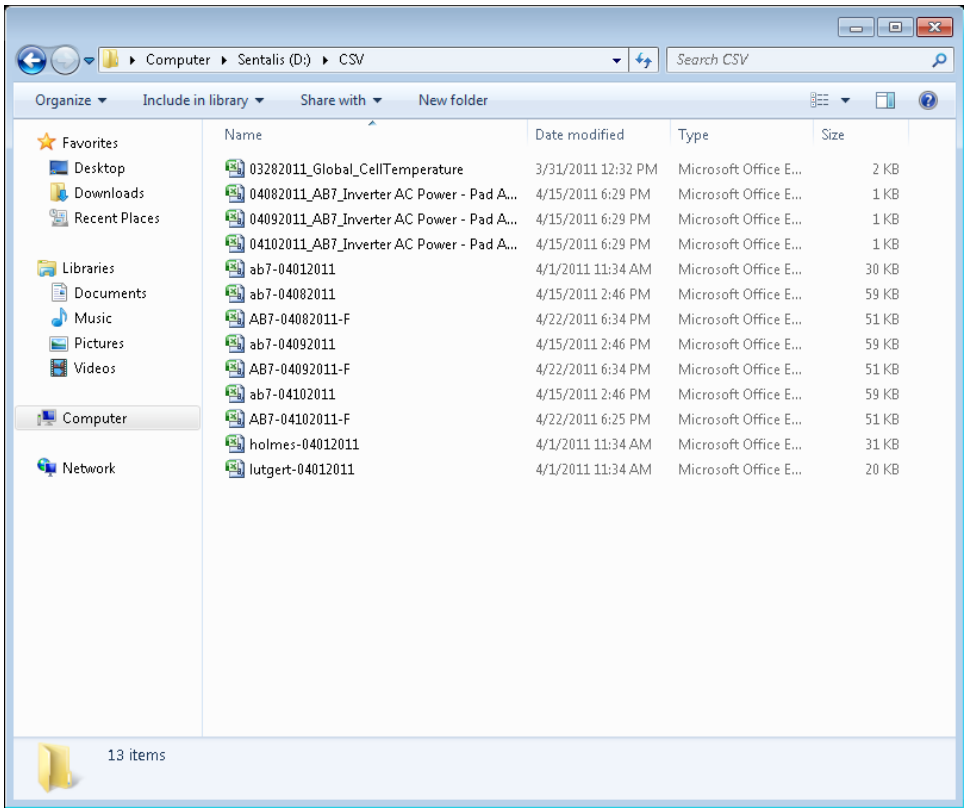

Computer ▸ Sentalis (D:) ▸ CSV
Search CSV
Organize
Include in library
Share with
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Network
Name
Date modified
Type
Size
03282011_Global_CellTemperature 3/31/2011 12:32 PM Microsoft Office E... 2 KB
04082011_AB7_Inverter AC Power - Pad A... 4/15/2011 6:29 PM Microsoft Office E... 1 KB
04092011_AB7_Inverter AC Power - Pad A... 4/15/2011 6:29 PM Microsoft Office E... 1 KB
04102011_AB7_Inverter AC Power - Pad A... 4/15/2011 6:29 PM Microsoft Office E... 1 KB
ab7-04012011 4/1/2011 11:34 AM Microsoft Office E... 30 KB
ab7-04082011 4/15/2011 2:46 PM Microsoft Office E... 59 KB
AB7-04082011-F 4/22/2011 6:34 PM Microsoft Office E... 51 KB
ab7-04092011 4/15/2011 2:46 PM Microsoft Office E... 59 KB
AB7-04092011-F 4/22/2011 6:34 PM Microsoft Office E... 51 KB
ab7-04102011 4/15/2011 2:46 PM Microsoft Office E... 59 KB
AB7-04102011-F 4/22/2011 6:25 PM Microsoft Office E... 51 KB
holmes-04012011 4/1/2011 11:34 AM Microsoft Office E... 31 KB
lutgert-04012011 4/1/2011 11:34 AM Microsoft Office E... 20 KB
13 items

Figure 4.2 Downloaded CSV Files

4.3 Installation

4.3.1 GParted

In order to set up the eBox to run Ubuntu we need to format the compact flash card correctly. This is done by downloading the latest stable release of GParted [6]. Once the Ubuntu Minimal Install image is downloaded we will use UNetBootIn [5] in order to flash the ISO to a USB flash

drive (at least 1gb capacity). The USB flash drive needs to be formatted in Fat32 using the HPDriveKey [8] utility. Figure 4.3.

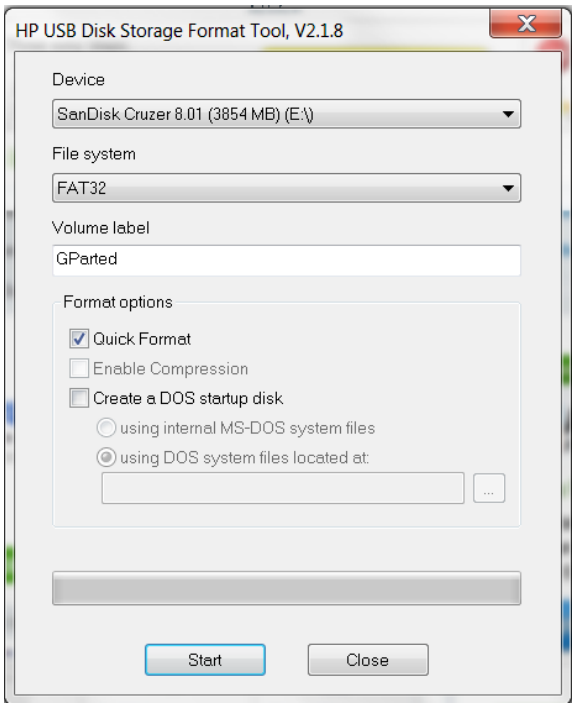


Figure (4.3) – Format USB flash drive

Once the USB flash drive is formatted, open up UNetBootIn and set to the following settings.

Figure 4.4

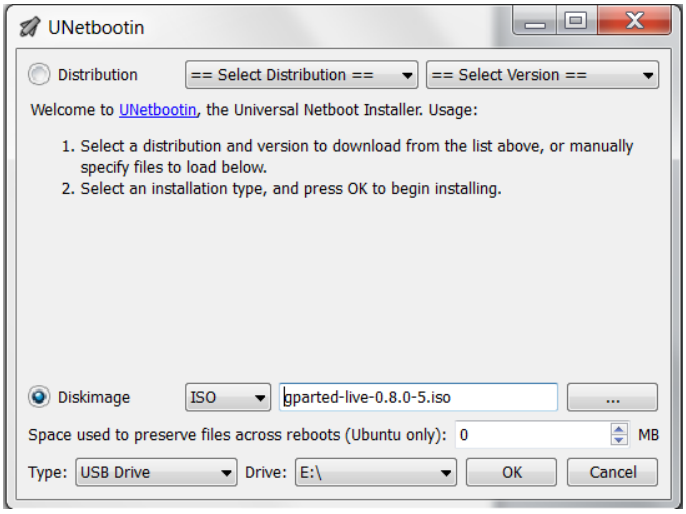


Figure (4.4) – Burn ISO to the USB flash drive

When you hit ok it will start the burn process, once complete it is ready to be ran on the eBox's compact flash card.

On boot go into the BIOS using DEL and set to boot from the GParted USB flash drive. Then restart the eBox. When the eBox boots from the USB follow these instructions:

1. Enter
2. Enter
3. Enter
4. Delete all partitions
5. On unallocated partition, create a new partition with max allowed space at ext2 file system.
6. Click add

7. Apply

8. Apply

9. Once partition is created, close

10. Turn off eBox.

4.3.2 Ubuntu

Now that the compact flash card is formatted with a single partition we are ready to install Ubuntu. However, we need to first create the Ubuntu USB flash drive. The instructions are similar to that of creating the GParted USB. Download the minimal install ISO from Ubuntu's website [9]. Format the USB flash drive (at least 1gb of capacity) using HPDriveKey [8] utility.

Figure 4.5

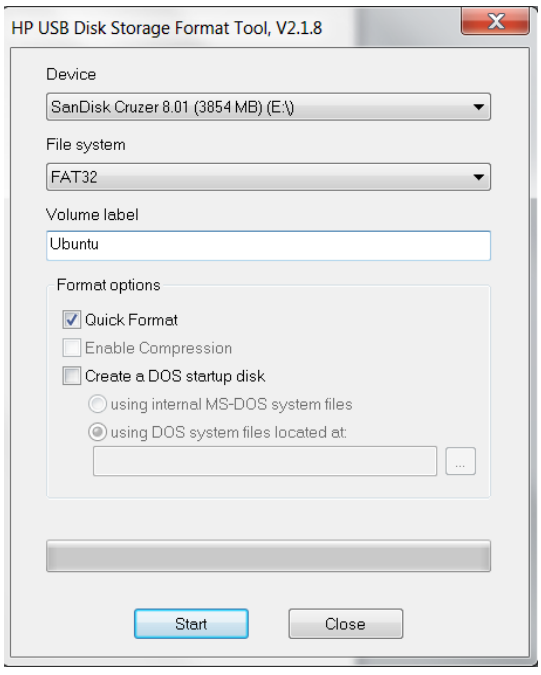


Figure (4.5) – Format USB flash drive

Once the USB flash drive is formatted at FAT32 open UnetBootIn and set to the following settings. Figure 4.6.

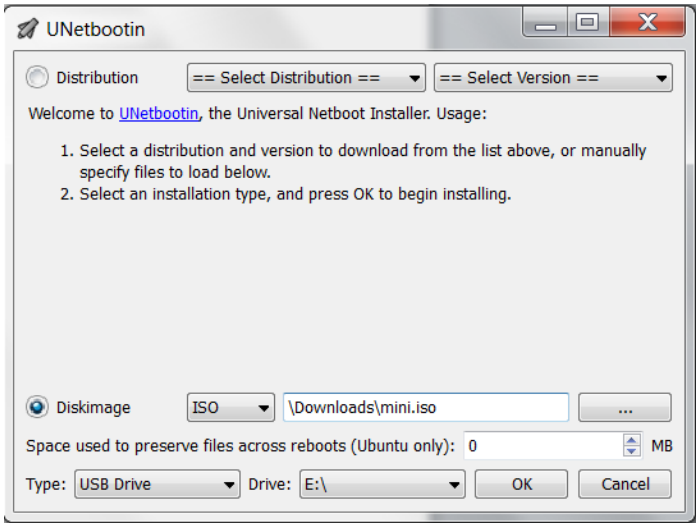


Figure (4.6) – Burn ISO to USB flash drive

Once UNetBootIn has finished burning the ISO to the USB flash drive insert the USB into the eBox and set the eBox to boot from this USB in the BIOS. When the Ubuntu setup starts follow these instructions:

1. Command line install
2. Enter
3. Enter
4. Enter
5. Enter
6. Enter

7. Set name to davinci
8. Continue
9. Enter
10. Enter
11. Continue
12. Set time zone settings
13. Yes
14. Guided
15. Use entire Disk
16. Set to the Compact Flash card
17. Continue
18. Yes
19. Create user name and password
20. No
21. Select no for automatic updates
22. Grub
23. Yes

24. Yes

Ubuntu will install. Once the installation has completed, restart the eBox.

4.3.3 Samba

Once the eBox has restarted and loaded into Ubuntu we need to install Samba. This is done from command line by typing

```
sudo apt-get install samba
```

Once Samba is installed we need to set up Samba to share a local folder. Now we need to edit the Samba config file. sudo gedit /etc/samba/smb.conf

Find this section in the file:

```
####### Authentication #######

# "security = user" is always a good idea. This will require a Unix account
# in this server for every user accessing the server. See
# /usr/share/doc/samba-doc/htmldocs/Samba-HOWTO-Collection/ServerType.html
# in the samba-doc package for details.
;  security = user
```

Uncomment the security line, and add another line to make it look like this:

```
security = user
username map = /etc/samba/smbusers
```

This will set Samba to use the smbusers file for looking up the user list.

Create a Samba User

There are two steps to creating a user. First we'll run the smbpasswd utility to create a samba password for the user.

```
sudo smbpasswd -a <username>
```

Next, we'll add that username to the smbusers file.

```
sudo gedit /etc/samba/smbusers
```

Add in the following line, substituting the username with the one you want to give access to. The format is <ubuntuusername> = "<samba username>". You can use a different samba user name to map to an ubuntu account, but that's not really necessary right now.

<username> = "<username>"

Now you can create samba shares and give access to the users that you listed here.

To share the home directories, open up smb.conf with the following command:

sudo gedit /etc/samba/smb.conf

Find this section of the file, and make it match the following:

```
#======================= Share Definitions =======================

# Un-comment the following (and tweak the other settings below to suit)
# to enable the default home directory shares. This will share each
# user's home directory as \\server\username
[homes]
comment = Home Directories
browseable = yes

# By default, \\server\username shares can be connected to by anyone
# with access to the samba server. Un-comment the following parameter
# to make sure that only "username" can connect to \\server\username
valid users = %S

# By default, the home directories are exported read-only. Change next
# parameter to 'yes' if you want to be able to write to them.
writable = yes
```

Now you should be able to map a drive on windows using the following share format:

\\davinci\sharedavinci

For example, if the Ubuntu machine is named davinci, and the username is sharedavinci, your share path would be \\davinci\sharedavinci

4.3.4 Mount the Backup Server's shared drive

In order to mount a shared windows drive the following needs to be done:

Create the Mount Point

Any time you want to map a drive in Linux, you should create the folder first. In this case, I want to access that shared folder from /mnt/win, so I'll first need to create the folder.

```
sudo mkdir /mnt/win
```

Note that I'm using sudo because I'm creating the folder in a system-wide location owned by root.

Actually Mounting the Shared Folder

You can run the following command to mount the shared folder to a local path:

```
mount –t cifs –o username=<username>,password=<password> //backupserver/backups /mnt/win
```

Note that you'd want to use your own details in the command. At this point you should be able to access the shared folder from /media/Video

Now we have created the link to the shared windows drive.

4.3.5 Java and Sentalis Fetch CSV Jar

In order to get the Sentalis Fetch CSV Jar to run correctly on daVinci we need to set up the Java

Runtime Environment. This is done with the following command:

```
sudo apt-get install openjdk-6-jre
```

Once this is installed we are ready to run the Jar.

4.4 Source Code

To run the source code, the following library needs to be added to the classpath: htmlunit. After that all there is left is to compile and run the program.

See the appendix for source code.

When the program runs, daVinci will download the CSV files from the Draker-Labs website and place them locally and on the shared drive for the main server. It will run once every hour and update the files that are saved.

5. Conclusion

The daVinci Sentalis Fetch CSV server was designed as a redundant data logger for the Senior Software Engineering Group project FINEMAN. Its purpose is to be a fallback for the main server in case any issues were to arise. Since the CSV files are the catalyst for the remaining of the project it was deemed necessary for the CSV files to be downloaded and stored separately from the main server but also store backups on the main server in case of its own failure.

This level of redundancy will allow for future projects to continue work where we left off and elaborate on the main project without affecting the CSV download process. When dealing with real world examples such as data loggers it has been shown that redundancy is the key to success.

6. References

[1] eBox-4864 User Manual, May, 2008, http://www.compactpc.com.tw/files/eBox-4_Manual.zip

[2] Eclipse Classic 3.6.1 IDE, March, 2011, http://www.eclipse.org/

[3] Java Runtime Environment Version 6 Update 24, March, 2011

http://java.com/en/download/chrome.jsp?locale=en&host=java.com

[4] HTMLUnit, August, 2010, http://htmlunit.sourceforge.net/

[5] UNetBootIn, 2011, http://unetbootin.sourceforge.net/

[6] GParted, March, 2011, http://gparted.sourceforge.net/

[7] Samba, 2011, http://www.samba.org/

[8] HP USB Disk Storage Format Tool, 2011, http://download.cnet.com/HP-USB-Disk-Storage-Format-Tool/3000-2094_4-10974082.html

[9] Ubuntu Minimal Install, 2011,

http://archive.ubuntu.com/ubuntu/dists/maverick/main/installer-i386/current/images/netboot/mini.iso

[10] SolarEMS Website, Sentalis, 2011, http://www.solarems.net

Appendix

Source Code:

```
//import Java libraries
import java.io.*;
import java.net.URL;
import java.util.*;
import java.security.GeneralSecurityException;
import java.text.*;
import javax.management.timer.Timer;
//import HtmlUnit libraries
import com.gargoylesoftware.htmlunit.BrowserVersion;
import com.gargoylesoftware.htmlunit.Page;
import com.gargoylesoftware.htmlunit.RefreshHandler;
import com.gargoylesoftware.htmlunit.WebClient;
import com.gargoylesoftware.htmlunit.html.HtmlForm;
import com.gargoylesoftware.htmlunit.html.HtmlPage;
import com.gargoylesoftware.htmlunit.TextPage;

/**
 * Senior Software Engineering Group Project: Log in to Sentalis and download
 * CSV file Authors: Chris Steiner and *Justin Hodnett* (*Previous GUI Interface)
 * Last Edited: April 6, 2011
 * Version: 2.0
 *
 * Note: This is a final product for the Senior Software Engineering Group
 * Project involving logging into the Sentalis website to download a CSV file to
 * be used by other groups. This program is not a commercial product and
 * therefore does not come with any warranties. Some tweaks may need to be made,
 * such as the time to wait for downloading a file.
 *
 */
public class SentalisFetchCSV {
	// calls class fetchCSV
	public static void main(String[] args) throws Exception {
		fetchCSV();
	}

	// Fetching the CSV
	private static void fetchCSV()
			throws Exception {
		// Create and initialize WebClient object
		WebClient webClient = new WebClient(BrowserVersion.FIREFOX_2);
		webClient.setThrowExceptionOnScriptError(false);
		webClient.setRefreshHandler(new RefreshHandler() {
			public void handleRefresh(Page page, URL url, int arg)
					throws IOException {
				System.out.println("handleRefresh");
			}

		});

		// Ignore SSL certificates
		try {
			webClient.setUseInsecureSSL(true);
		} catch (GeneralSecurityException ex) {
			System.out
					.println("Failed setting ignore SSL on HTMLUnit Web
Client: "
							+ ex);
```

```
                    fetchCSV();
              }

              // visit Sentalis login page and get the Form object
              HtmlPage page = (HtmlPage) webClient.getPage("https://solarems.net");
              HtmlForm form = (HtmlForm) page.getElementById("new_user_session");

              // Enter email and password

       form.getInputByName("user_session[email]").setValueAttribute("zalewski@fgcu.edu
");
              form.getInputByName("user_session[password]").setValueAttribute(
                           "Draker-Labs");

              // Click "Login In" button/link
              page = (HtmlPage) form.getInputByValue("Log In").click();

              // setup date information for dynamic link and output file
              DateFormat dayFormat = new SimpleDateFormat("dd");
              DateFormat monthFormat = new SimpleDateFormat("MM");
              DateFormat yearFormat = new SimpleDateFormat("yyyy");
              Date day = new Date();
              Date month = new Date();
              Date year = new Date();

              TextPage ab7 = null;
              TextPage holmes = null;
              TextPage lutgert = null;
              // submit the link for the .csv file for ab7
              try {
                    ab7 = (TextPage) webClient
                                 .getPage("https://solarems.net/projects/36-fgcu-
ab7/data.csv?date_range="
                                              + monthFormat.format(month)
                                              + "%2F"
                                              + dayFormat.format(day)
                                              + "%2F"
                                              + yearFormat.format(year)
                                              +
"&v%5B%5D=5208&v%5B%5D=6002&v%5B%5D=6003&v%5B%5D=3031&v%5B%5D=5959&v%5B%5D=5999&v%5B%
5D=3029&v%5B%5D=3028&v%5B%5D=3033&v%5B%5D=3032&v%5B%5D=3036&v%5B%5D=3034&v%5B%5D=3035&v
%5B%5D=3119&v%5B%5D=3114&v%5B%5D=3115&v%5B%5D=3112&v%5B%5D=3113&v%5B%5D=3116&v%5B%5D=3
118&v%5B%5D=3108&v%5B%5D=3120&v%5B%5D=3109&v%5B%5D=3110&v%5B%5D=3111&v%5B%5D=3107&v%5B
%5D=3046&v%5B%5D=3044&v%5B%5D=3045&v%5B%5D=3042&v%5B%5D=3043&v%5B%5D=5142&v%5B%5D=3021
&v%5B%5D=3048&v%5B%5D=3038&v%5B%5D=3047&v%5B%5D=3039&v%5B%5D=3040&v%5B%5D=3041&v%5B%5D
=3037&v%5B%5D=5963&v%5B%5D=5967&v%5B%5D=5964&v%5B%5D=3023&v%5B%5D=3016&v%5B%5D=3019&v%
5B%5D=3015&v%5B%5D=3017&v%5B%5D=3018&v%5B%5D=3025&v%5B%5D=3026&v%5B%5D=3027&v%5B%5D=30
20&v%5B%5D=3071&v%5B%5D=3065&v%5B%5D=3068&v%5B%5D=3063&v%5B%5D=3064&v%5B%5D=5164&v%5B%
5D=3066&v%5B%5D=3070&v%5B%5D=3059&v%5B%5D=3069&v%5B%5D=3060&v%5B%5D=3061&v%5B%5D=3062&
v%5B%5D=3058&v%5B%5D=5976&v%5B%5D=5980&v%5B%5D=5977&v%5B%5D=3050&v%5B%5D=3053&v%5B%5D=
3049&v%5B%5D=3051&v%5B%5D=3052&v%5B%5D=3055&v%5B%5D=3056&v%5B%5D=3057&v%5B%5D=3054");

                    byte buf[] = new byte[1024];
                    int len;
                    File of = new File("/home/sharedavinci/csv/ab7-"
                                 + monthFormat.format(month) + dayFormat.format(day)
                                 + yearFormat.format(year) + ".csv");
                    OutputStream os = new FileOutputStream(of);

                    // Get the input stream
                    InputStream is = ab7.getWebResponse().getContentAsStream();

                    // Try to read data in and write it to a file
```

```
                    while ((len = is.read(buf)) > 0) {
                          os.write(buf, 0, len);
                    }
                    os.flush();
                    os.close();
                    is.close();
              } catch (ClassCastException e) {
                    fetchCSV();
              }
              //make backup of .csv file for ab7
              try {
                    ab7 = (TextPage) webClient
                                .getPage("https://solarems.net/projects/36-fgcu-
ab7/data.csv?date_range="
                                            + monthFormat.format(month)
                                            + "%2F"
                                            + dayFormat.format(day)
                                            + "%2F"
                                            + yearFormat.format(year)
                                            +
"&v%5B%5D=5208&v%5B%5D=6002&v%5B%5D=6003&v%5B%5D=3031&v%5B%5D=5959&v%5B%5D=5999&v%5B%5
D=3029&v%5B%5D=3028&v%5B%5D=3033&v%5B%5D=3032&v%5B%5D=3036&v%5B%5D=3034&v%5B%5D=3035&v
%5B%5D=3119&v%5B%5D=3114&v%5B%5D=3115&v%5B%5D=3112&v%5B%5D=3113&v%5B%5D=3116&v%5B%5D=3
118&v%5B%5D=3108&v%5B%5D=3120&v%5B%5D=3109&v%5B%5D=3110&v%5B%5D=3111&v%5B%5D=3107&v%5B
%5D=3046&v%5B%5D=3044&v%5B%5D=3045&v%5B%5D=3042&v%5B%5D=3043&v%5B%5D=5142&v%5B%5D=3021
&v%5B%5D=3048&v%5B%5D=3038&v%5B%5D=3047&v%5B%5D=3039&v%5B%5D=3040&v%5B%5D=3041&v%5B%5D
=3037&v%5B%5D=5963&v%5B%5D=5967&v%5B%5D=5964&v%5B%5D=3023&v%5B%5D=3016&v%5B%5D=3019&v%
5B%5D=3015&v%5B%5D=3017&v%5B%5D=3018&v%5B%5D=3025&v%5B%5D=3026&v%5B%5D=3027&v%5B%5D=30
20&v%5B%5D=3071&v%5B%5D=3065&v%5B%5D=3068&v%5B%5D=3063&v%5B%5D=3064&v%5B%5D=5164&v%5B%
5D=3066&v%5B%5D=3070&v%5B%5D=3059&v%5B%5D=3069&v%5B%5D=3060&v%5B%5D=3061&v%5B%5D=3062&
v%5B%5D=3058&v%5B%5D=5976&v%5B%5D=5980&v%5B%5D=5977&v%5B%5D=3050&v%5B%5D=3053&v%5B%5D=
3049&v%5B%5D=3051&v%5B%5D=3052&v%5B%5D=3055&v%5B%5D=3056&v%5B%5D=3057&v%5B%5D=3054");

                    byte buf[] = new byte[1024];
                    int len;
                    File of = new File("/mnt/win/backups/ab7-"
                                + monthFormat.format(month) + dayFormat.format(day)
                                + yearFormat.format(year) + ".csv");
                    OutputStream os = new FileOutputStream(of);

                    // Get the input stream
                    InputStream is = ab7.getWebResponse().getContentAsStream();

                    // Try to read data in and write it to a file
                    while ((len = is.read(buf)) > 0) {
                          os.write(buf, 0, len);
                    }
                    os.flush();
                    os.close();
                    is.close();
              } catch (ClassCastException e) {
                    fetchCSV();
              }

              // submit the link for the .csv file for holmes
              try {
                    holmes = (TextPage) webClient
                                .getPage("https://solarems.net/projects/37-fgcu-
holmes/data.csv?date_range="
                                            + monthFormat.format(month)
                                            + "%2F"
                                            + dayFormat.format(day)
                                            + "%2F"
```

```
                                                + yearFormat.format(year)
                                                +
"&v%5B%5D=5280&v%5B%5D=5293&v%5B%5D=5294&v%5B%5D=3124&v%5B%5D=3123&v%5B%5D=5282&v%5B%5
D=3122&v%5B%5D=3121&v%5B%5D=3126&v%5B%5D=3125&v%5B%5D=3129&v%5B%5D=3127&v%5B%5D=3128&v
%5B%5D=3142&v%5B%5D=3137&v%5B%5D=3138&v%5B%5D=3135&v%5B%5D=3136&v%5B%5D=3139&v%5B%5D=3
141&v%5B%5D=3131&v%5B%5D=3143&v%5B%5D=3132&v%5B%5D=3133&v%5B%5D=3134&v%5B%5D=3130&v%5B
%5D=3168&v%5B%5D=3162&v%5B%5D=3163&v%5B%5D=3160&v%5B%5D=3161&v%5B%5D=5236&v%5B%5D=3164
&v%5B%5D=3167&v%5B%5D=3156&v%5B%5D=3166&v%5B%5D=3157&v%5B%5D=3158&v%5B%5D=3159&v%5B%5D
=3155&v%5B%5D=5249&v%5B%5D=5237&v%5B%5D=5250&v%5B%5D=3150&v%5B%5D=3145&v%5B%5D=3148&v%
5B%5D=3144&v%5B%5D=3146&v%5B%5D=3147&v%5B%5D=3152&v%5B%5D=3153&v%5B%5D=3154&v%5B%5D=31
49&v%5B%5D=3185&v%5B%5D=3184&v%5B%5D=3193&v%5B%5D=3180&v%5B%5D=3182&v%5B%5D=5258&v%5B%
5D=3186&v%5B%5D=3183&v%5B%5D=3188&v%5B%5D=3181&v%5B%5D=3189&v%5B%5D=3190&v%5B%5D=3191&
v%5B%5D=3187&v%5B%5D=5271&v%5B%5D=5259&v%5B%5D=5272&v%5B%5D=3175&v%5B%5D=3171&v%5B%5D=
3174&v%5B%5D=3169&v%5B%5D=3170&v%5B%5D=3173&v%5B%5D=3177&v%5B%5D=3178&v%5B%5D=3179&v%5
B%5D=3172");

                    byte buf[] = new byte[1024];
                    int len;
                    File of = new File("/home/sharedavinci/csv/holmes-"
                                + monthFormat.format(month) + dayFormat.format(day)
                                + yearFormat.format(year) + ".csv");
                    OutputStream os = new FileOutputStream(of);

                    // Get the input stream
                    InputStream is = holmes.getWebResponse().getContentAsStream();

                    // Try to read data in and write it to a file
                    while ((len = is.read(buf)) > 0) {
                          os.write(buf, 0, len);
                    }
                    os.flush();
                    os.close();
                    is.close();
              } catch (ClassCastException e) {
                    fetchCSV();
              }
              // create backup .csv file for holmes
              try {
                    holmes = (TextPage) webClient
                                .getPage("https://solarems.net/projects/37-fgcu-
holmes/data.csv?date_range="
                                                + monthFormat.format(month)
                                                + "%2F"
                                                + dayFormat.format(day)
                                                + "%2F"
                                                + yearFormat.format(year)
                                                +
"&v%5B%5D=5280&v%5B%5D=5293&v%5B%5D=5294&v%5B%5D=3124&v%5B%5D=3123&v%5B%5D=5282&v%5B%5
D=3122&v%5B%5D=3121&v%5B%5D=3126&v%5B%5D=3125&v%5B%5D=3129&v%5B%5D=3127&v%5B%5D=3128&v
%5B%5D=3142&v%5B%5D=3137&v%5B%5D=3138&v%5B%5D=3135&v%5B%5D=3136&v%5B%5D=3139&v%5B%5D=3
141&v%5B%5D=3131&v%5B%5D=3143&v%5B%5D=3132&v%5B%5D=3133&v%5B%5D=3134&v%5B%5D=3130&v%5B
%5D=3168&v%5B%5D=3162&v%5B%5D=3163&v%5B%5D=3160&v%5B%5D=3161&v%5B%5D=5236&v%5B%5D=3164
&v%5B%5D=3167&v%5B%5D=3156&v%5B%5D=3166&v%5B%5D=3157&v%5B%5D=3158&v%5B%5D=3159&v%5B%5D
=3155&v%5B%5D=5249&v%5B%5D=5237&v%5B%5D=5250&v%5B%5D=3150&v%5B%5D=3145&v%5B%5D=3148&v%
5B%5D=3144&v%5B%5D=3146&v%5B%5D=3147&v%5B%5D=3152&v%5B%5D=3153&v%5B%5D=3154&v%5B%5D=31
49&v%5B%5D=3185&v%5B%5D=3184&v%5B%5D=3193&v%5B%5D=3180&v%5B%5D=3182&v%5B%5D=5258&v%5B%
5D=3186&v%5B%5D=3183&v%5B%5D=3188&v%5B%5D=3181&v%5B%5D=3189&v%5B%5D=3190&v%5B%5D=3191&
v%5B%5D=3187&v%5B%5D=5271&v%5B%5D=5259&v%5B%5D=5272&v%5B%5D=3175&v%5B%5D=3171&v%5B%5D=
3174&v%5B%5D=3169&v%5B%5D=3170&v%5B%5D=3173&v%5B%5D=3177&v%5B%5D=3178&v%5B%5D=3179&v%5
B%5D=3172");

                    byte buf[] = new byte[1024];
                    int len;
```

```
                    File of = new File("/mnt/win/backups/holmes-"
                                + monthFormat.format(month) + dayFormat.format(day)
                                + yearFormat.format(year) + ".csv");
                    OutputStream os = new FileOutputStream(of);

                    // Get the input stream
                    InputStream is = holmes.getWebResponse().getContentAsStream();

                    // Try to read data in and write it to a file
                    while ((len = is.read(buf)) > 0) {
                          os.write(buf, 0, len);
                    }
                    os.flush();
                    os.close();
                    is.close();
              } catch (ClassCastException e) {
                    fetchCSV();
              }

              // submit the link for the .csv file for lutgert
              try {
                    lutgert = (TextPage) webClient
                                .getPage("https://solarems.net/projects/38-fgcu-
lutgert/data.csv?date_range="
                                          + monthFormat.format(month)
                                          + "%2F"
                                          + dayFormat.format(day)
                                          + "%2F"
                                          + yearFormat.format(year)
                                          +
"&v%5B%5D=5918&v%5B%5D=5318&v%5B%5D=5919&v%5B%5D=3197&v%5B%5D=3196&v%5B%5D=5316&v%5B%5
D=3195&v%5B%5D=3194&v%5B%5D=3199&v%5B%5D=3198&v%5B%5D=3202&v%5B%5D=3200&v%5B%5D=3201&v
%5B%5D=3213&v%5B%5D=3210&v%5B%5D=3211&v%5B%5D=3208&v%5B%5D=3209&v%5B%5D=3215&v%5B%5D=3
212&v%5B%5D=3204&v%5B%5D=3214&v%5B%5D=3205&v%5B%5D=3206&v%5B%5D=3207&v%5B%5D=3203&v%5B
%5D=5930&v%5B%5D=5927&v%5B%5D=5928&v%5B%5D=5925&v%5B%5D=5926&v%5B%5D=5932&v%5B%5D=5931
&v%5B%5D=5929&v%5B%5D=5914&v%5B%5D=5958&v%5B%5D=5922&v%5B%5D=5923&v%5B%5D=5924&v%5B%5D
=5913&v%5B%5D=5312&v%5B%5D=5308&v%5B%5D=5915&v%5B%5D=3223&v%5B%5D=3218&v%5B%5D=3221&v%
5B%5D=3217&v%5B%5D=3219&v%5B%5D=3220&v%5B%5D=3225&v%5B%5D=3226&v%5B%5D=3227&v%5B%5D=32
22");

                    byte buf[] = new byte[1024];
                    int len;
                    File of = new File("/home/sharedavinci/csv/lutgert-"
                                + monthFormat.format(month) + dayFormat.format(day)
                                + yearFormat.format(year) + ".csv");
                    OutputStream os = new FileOutputStream(of);

                    // Get the input stream
                    InputStream is = lutgert.getWebResponse().getContentAsStream();

                    // Try to read data in and write it to a file
                    while ((len = is.read(buf)) > 0) {
                          os.write(buf, 0, len);
                    }
                    os.flush();
                    os.close();
                    is.close();
              } catch (ClassCastException e) {
                    fetchCSV();
              }
              // create backup .csv file for lutgert
              try {
                    lutgert = (TextPage) webClient
```

```
                                    .getPage("https://solarems.net/projects/38-fgcu-
lutgert/data.csv?date_range="
                                                + monthFormat.format(month)
                                                + "%2F"
                                                + dayFormat.format(day)
                                                + "%2F"
                                                + yearFormat.format(year)
                                                +
"&v%5B%5D=5918&v%5B%5D=5318&v%5B%5D=5919&v%5B%5D=3197&v%5B%5D=3196&v%5B%5D=5316&v%5B%5
D=3195&v%5B%5D=3194&v%5B%5D=3199&v%5B%5D=3198&v%5B%5D=3202&v%5B%5D=3200&v%5B%5D=3201&v
%5B%5D=3213&v%5B%5D=3210&v%5B%5D=3211&v%5B%5D=3208&v%5B%5D=3209&v%5B%5D=3215&v%5B%5D=3
212&v%5B%5D=3204&v%5B%5D=3214&v%5B%5D=3205&v%5B%5D=3206&v%5B%5D=3207&v%5B%5D=3203&v%5B
%5D=5930&v%5B%5D=5927&v%5B%5D=5928&v%5B%5D=5925&v%5B%5D=5926&v%5B%5D=5932&v%5B%5D=5931
&v%5B%5D=5929&v%5B%5D=5914&v%5B%5D=5958&v%5B%5D=5922&v%5B%5D=5923&v%5B%5D=5924&v%5B%5D
=5913&v%5B%5D=5312&v%5B%5D=5308&v%5B%5D=5915&v%5B%5D=3223&v%5B%5D=3218&v%5B%5D=3221&v%
5B%5D=3217&v%5B%5D=3219&v%5B%5D=3220&v%5B%5D=3225&v%5B%5D=3226&v%5B%5D=3227&v%5B%5D=32
22");

                        byte buf[] = new byte[1024];
                        int len;
                        File of = new File("/mnt/win/backups/lutgert-"
                                    + monthFormat.format(month) + dayFormat.format(day)
                                    + yearFormat.format(year) + ".csv");
                        OutputStream os = new FileOutputStream(of);

                        // Get the input stream
                        InputStream is = lutgert.getWebResponse().getContentAsStream();

                        // Try to read data in and write it to a file
                        while ((len = is.read(buf)) > 0) {
                              os.write(buf, 0, len);
                        }
                        os.flush();
                        os.close();
                        is.close();
                  } catch (ClassCastException e) {
                        fetchCSV();
                  }
                  //sleeping for one hour
                  System.out.println("Waiting one hour...");
                  Thread.sleep(Timer.ONE_HOUR);
                  //download again
                  System.out.println("Downloading CSV Files...");
                  fetchCSV();
            }
}
```